REMARKS

Reconsideration and allowance are respectfully requested in view of the foregoing amendments and the following remarks.

Claims 1-17 and 19-23 are pending. Claims 1, 8, 13 and 21 were amended to more clearly describe the claimed invention. Claim 18 was canceled without prejudice or disclaimer. Applicants submit that the amendments to the claims do not narrow the scope of the claims.

Interview of January 24, 2006

Applicants wish to thank the Examiner and his SPE for the interview on January 24, 2006, in which Applicants' representative explained how the claims differ from U.S. Patent No. 6,651,101 Gai et al. ("Gai"). Applicants further thank the Examiner for agreeing to consider this amendment.

Rejection of Claims 1, 6, 7, 13, 19-21 and 23

On page 2 of the non-Final Office Action, the Examiner rejected claims 1, 6, 7, 13, 19- 21 and 23 under 35 U.S.C. 102(e) as allegedly being anticipated by U.S. Patent No. 6,651,101 to Gai et al. ("Gai"). Applicants respectfully traverse the rejection.

Independent claim 1 is directed to a communication protocol that includes, among other things, a first utility program adding a token, a first category type identifier corresponding to a first data type, and a first data type identifier corresponding to the first data type, to data to form an information packet and then, transparent to a sending application, using the transport mechanism to transmit the information packet to a second computer system.

Gai discloses a method and apparatus for identifying network data traffic flows and for applying quality of service or policy treatments thereto (Gai, at col. 1, lines 23-26). Fig. 2

of Gai discloses an exemplary computer network. Host/server 222 of Gai includes an application program 224, a flow declaration component 226 and a communication facility 228. Flow declaration component 226 includes a message generator 230 in communicating relation with communication facility 228 (Gai, at col. 6, lines 11-15). Flow declaration component 226 is coupled to a memory 232 for storing one or more traffic flow data structures (Gai, at col. 6, lines 15-17). Application program 224 is arranged to communicate with communications facility 228 and through an Application Program Interface (API) layer 236 to flow declaration component 226 (Gai, at col. 6, lines 17-20).

Application program 224 can communicate with an end station 212 across a network 200 via communication facility 228 at host/server 222 (Gai, at col. 7, lines 56-59). Application program 224 may communicate with flow declaration component 226 through a number of API system calls to API layer 236 (Gai, at col. 7, lines 59-62). Generally, application program 224 issues API calls with one or more arguments which may be returned by flow declaration component 226 (Gai, at col. 7, lines 62-64).

On page 2 of the non-Final Office Action of December 29, 2005, the Examiner alleged that Gai, at col. 7, line 65 through col. 8, line 14, discloses a first utility program adding a token, a first category type identifier corresponding to a first data type, and a first data type identifier corresponding to the first data type, to data to form an information packet and then, transparently to a sending application, using the transport mechanism to transmit the information packet to a second computer system. Applicants disagree.

For a claim to be anticipated by a reference, the reference must disclose each and every feature of the claim. Gai fails to disclose each and every feature of claim 1.

Gai, at col. 7, line 65 through col. 8, line 14, discloses:

> In particular, upon initialization at host/server 222, the application program 224 preferably issues a StartUp( ) API call 410 to the API layer 236 at flow declaration component 226. Program 226 [sic] preferably loads the StartUp( ) call 410 with an application identifier that uniquely identifies application program 224 to component 226 as an argument. The application identifier

> may be a globally unique identifier (GUID), which is a 128 bit long value typically provided by the application developer, although other identifiers may also be used (e.g., application name). The StartUp( ) call 410 may be returned by the flow declaration component 226 with a version number as an argument. The version number corresponds to the version of software being executed by the flow declaration component 226. Other arguments, such as the quality-of-service (QoS) and/or traffic management resources that are available to traffic flows originating from program 224, may also be returned by flow declaration component 226.

Thus, Gai discloses application program 224 issuing a StartUp API call via API layer 236 (APL) at flow declaration component 226. Application program 224 loads the StartUp API call with an application identifier that uniquely identifies the application program to the flow control component. The StartUp call may return to application program 224 a software version number, as well as other arguments (see Fig. 4A).

Applicants submit that the StartUp API Call is made by the application program to introduce application program 224, identified by its unique application identifier, to flow declaration component 226 and for flow control component 226 to provide to the application program its software version, as well as other parameters, which can be returned to application program 224 as one or more arguments by flow declaration component 226.

Making an API call with an argument such as, for example, an application identifier from application program 224 to flow declaration component 226 via API layer 236, all within host/server 222 (see Gai, Fig. 2) is not equivalent to a first utility program adding a token to data to form an information data packet and then, transparently to the sending application, using a transport mechanism to transmit the information packet to a second computer system, as required by claim 1. One of ordinary skill in the art would understand that an information packet includes a header and data in a format to be communicated via a network (See Gai, at col. 2, lines 20-22, col. 3, lines 1-5 and Fig. 1B). An API call with arguments made from one process in a computer to another process in the same computer is not an information packet.

For at least the above reasons, Applicants submit that the cited portion of Gai, as well as any other portion of Gai, fails to disclose or suggest a first utility program adding a token, a first category type identifier corresponding to a first data type, to data to form an information packet and then, transparently to the sending application, using a transport mechanism to transmit the information packet to a second computer system, as required by claim 1.

Claim 1 further recites a second utility program, resident on the second computer system, using the token to locate the first data type identifier and the first category type identifier in the information packet. On page 3 of the non-Final Office Action of December 29, 2005, the Examiner argued that Gai, at col. 16, lines 21-47 discloses packets received at a policy enforcer are examined according to certain policy rules. Applicants disagree with the Examiner's analysis.

Gai, at col. 16, lines 21-47, discloses:

> As packets or frames are received at the local policy enforcer 210, they are examined by the packet/frame classifier 314. More specifically, the packet/frame classifier 314 parses the source and destination port fields 152, 154 (FIG. 1C) and the IP source and destination address fields 126, 128 and the protocol field 124 (FIG. 1B). This information is then supplied to the traffic flow state machine engine 310, which determines whether a traffic flow state has been established for such packets or frames. Assuming the packets or frames correspond to the anticipated flow from the program 224 to end station 212 (e.g., the IBM stock quote form), a traffic flow state will exist and have associated policy rules or service treatments as specified in the Policy Decision message 430 from policy server 216. Local policy enforcer 210 then applies the specified treatments to these packets or frames. For example, the traffic flow state machine engine 310 may instruct the packet/frame classifier, to set the DS field 132 (FIG. 1B) of such packets or frames to a value associated with best effort traffic. Similarly, the traffic flow state machine engine 310 may instruct the queue selector/mapping entity 318 to place these packets or frames in a particular (e.g., moderate priority) queue. Alternatively or in addition, packet/frame classifier may be instructed to load the ToS field 122 (FIG. 1B) or the user priority field 108 (FIG. 1A) with predetermined values so as to implement these treatments at other intermediate network devices, such as device 208.

Thus, Gai discloses a packet/frame classifier, included within a policy enforcer, examining received packets or frames. Source and destination port fields, source and destination IP

address fields, and a protocol field are parsed by the packet/frame classifier. The traffic flow state machine uses this information to determine whether a traffic flow state exists for the frames or packets. If a traffic flow state exists, then the traffic flow will have associated policy rules or service treatments, which will be applied by local policy enforcer to the frames or packets.

Applicants submit that the cited portion of Gai, or any other portion of Gai fails to disclose or suggest a second utility program, resident on a second computer system, using the token to locate the first data type identifier and the first category type identifier in the information packet, as required by claim 1. The source and destination port fields and the source and destination IP address fields are not the equivalent of a first data type identifier or a first category type identifier. Assuming *arguendo* that the protocol field includes a token, a point which Applicants do not concede, Gai fails to disclose or suggest using the token to locate the first data type identifier and the first category type identifier in the information packet. Therefore, Applicants submit that Gai fails to disclose or suggest a second utility program, resident on a second computer system, locating a first data type identifier and a first category type identifier in the information packet using a token, as required by claim 1.

On pages 7 and 8 of the non-Final Office Action, dated December 29, 2005, the Examiner stated that he

> interprets the first category identifier as a transaction type which Gai discloses as data information regarding user name, user department print and the first identifier as a subtransaction such as a print job on a HP laser jet printer.

The Examiner relied on Gai, col. 10, lines 8-25 for support. Applicants respectfully disagree with the Examiner.

Gai, at col. 10, lines 8-27, states:

> The application-level parameters may encompass a whole range of information relating to different aspects of the traffic flow from the application program 224. For example, application-level parameters include such information as user name (e.g., John Smith), user department (e.g., engineering, accounting, marketing, etc.), application name (e.g., SAP R/3,

> PeopleSoft, etc.), application module (e.g., SAP R/3 accounting form, SAP R/3 order entry form, etc.), transaction type (e.g., print), sub-transaction type (e.g., print on HP Laser Jet Printer), transaction name (e.g., print monthly sales report), sub-transaction name (e.g., print monthly sales report on A4 paper), application state (e.g., normal mode, critical mode, primary mode, back-up mode, etc.). For a video streaming application, the application-level parameters might include user name, film name, film compression method, film priority, optimal bandwidth, etc. Similarly, for a voice over IP application, the application-level parameters may include calling party, called party, compression method, service level of calling party (e.g., gold, silver, bronze), etc.

Thus, Gai discloses that data information regarding user name, user department, a subtransaction such as a print job on a HP laser jet printer are application-level parameters.

According to Gai, at col. 9, line 30 through col. 10, line 7, application-level parameters may be set by application program 224 issuing one or more "Set" API calls to flow declaration component 226 to cause the flow declaration component to load a traffic flow data structure with application-level parameters (also see Gai, at col. 8, lines 33-64 and Fig. 4A). That is, application program 224 issues one or more "Set" API calls via API layer 236 to flow declaration component 226, which sets the application-level parameters in the traffic flow data structure. As previously mentioned, application program 224 communicates with flow declaration component 226 via API layer 236 all on the same host/server 222. Therefore, no information packet and no network is used for communications between application program 224 and flow declaration component 226. Therefore, the Examiner's broad interpretation of data information, such as a user name, user department print and first identifier as a subtransaction, such as a print job on a HP laser printer as the first category identifier as a transaction type is irrelevant because Gai fails to disclose or suggest locating the first data type identifier and the first category type identifier in the information packet using the token. The user name, user department print and first identifier as a subtransaction, such as a print job on a HP laser printer as the first category identifier are simply **not included in an information packet**, as required by claim 1.

Because Gai fails to disclose or suggest each and every feature of claim 1, Applicants submit that claim 1 and dependent claims 6 and 7 are not anticipated by Gai. Applicants, therefore, respectfully request that the rejection of claims 1, 6 and 7 be withdrawn.

Claims 13, 21 and 23 recite features similar to those of claim 1 and are not anticipated by Gai for at least reasons similar to those discussed with respect to claim 1. Therefore, Applicants respectfully request that the rejection of independent claim 13, dependent claims 19 and 20, and independent claims 21 and 23 be withdrawn.

Rejection of Claims 2-5, 8-12, 14-18 and 22

On page 4 of the non-Final Office Action, the Examiner rejected claims 2-5, 8-12, 14-18 and 22 under 35 U.S.C. 103(a) as allegedly being unpatentable over Gai and further in view of U.S. Patent No. 6,654,786 to Fox et al. ("Fox"). Applicants respectfully traverse the rejection.

Claims 2-5, 14-18 and 22 depend either from claims 1, 13, or 21 which are not anticipated by Gai for at least the reasons provided with respect to claims 1, 13 and 21. Fox fails to satisfy the deficiencies of Gai. Therefore, Applicants respectfully request that the rejection of claims 2-5, 14-18 and 22 be withdrawn.

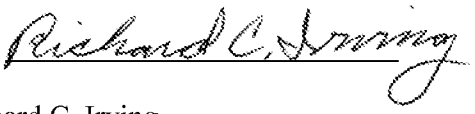
Claim 8 has features that are similar to those of claim 1. Therefore, Applicants submit that claim 8 is not anticipated by Gai for at least reasons similar to those provided with respect to claim 1. Fox fails to satisfy the deficiencies of Gai. Therefore, Applicants respectfully request that the rejection of claim 8 and dependent claims 9-12 be withdrawn.

CONCLUSION

Having addressed all rejections, Applicants respectfully submit that the subject application is in condition for allowance and a Notice to that effect is earnestly solicited.

Respectfully submitted,

Date: March 15, 2006

By: 

Correspondence Address:
Customer No. 49637

Richard C. Irving
Attorney for Applicants
Reg. No. 38,499
Phone: 410-414-3056
Fax No.: 410-510-1433